FaS 12차 세미나

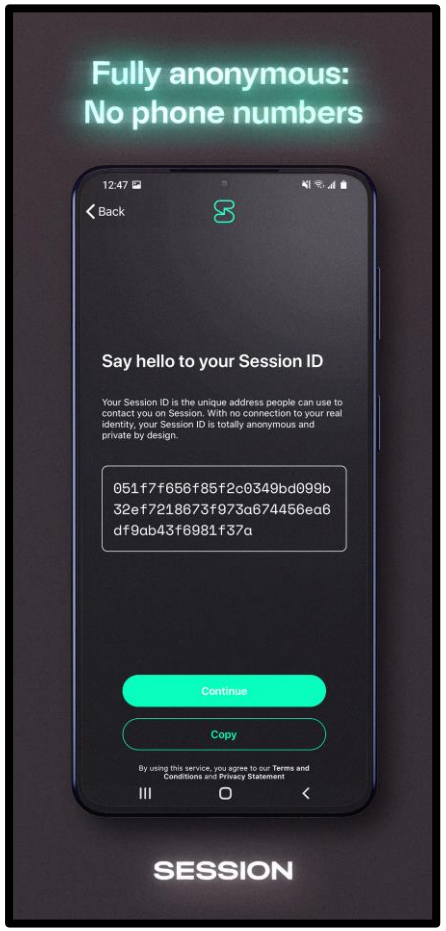
애플리케이션 분석 실습 -Session

20192233 박진철

CONTENTS

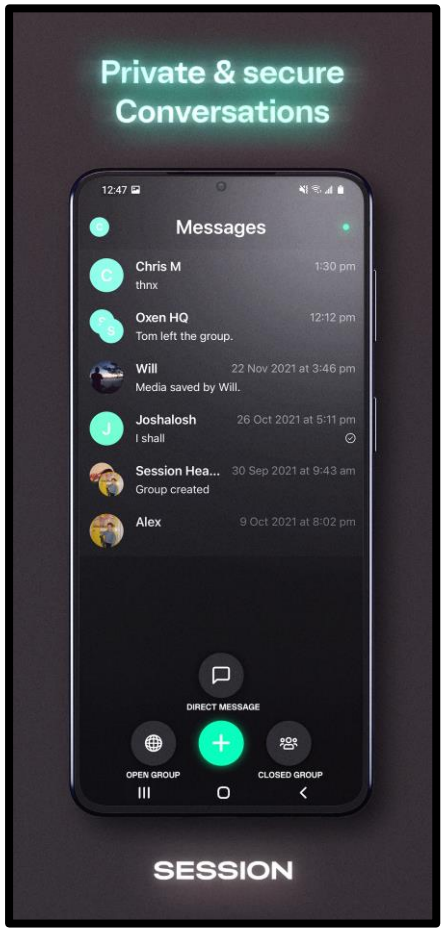
1. Session 소개

1. Session 소개



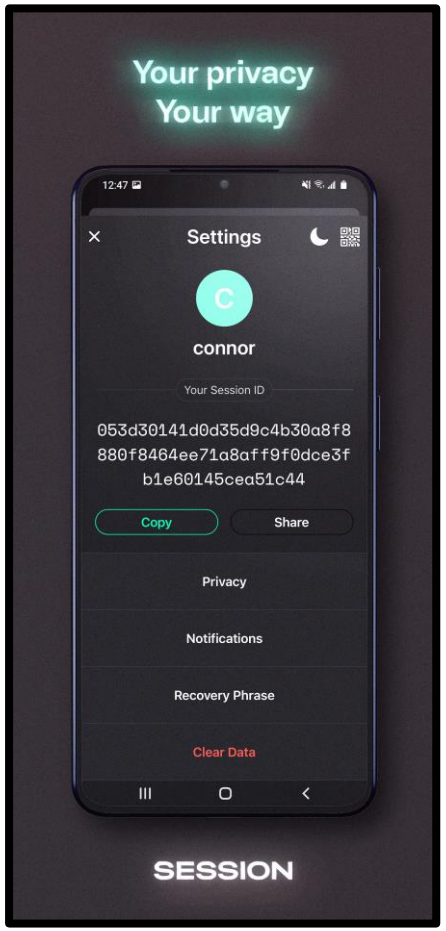
- Oxen Project에서 2020년 개발
- 익명 및 보안을 제공하는 개인 메신저
- 익명 계정생성, 첨부파일 보호 기능

1. Session 소개



- Oxen Project에서 2020년 개발
- 익명 및 보안을 제공하는 개인 메신저
- 익명 계정생성, 첨부파일 보호 기능

1. Session 소개

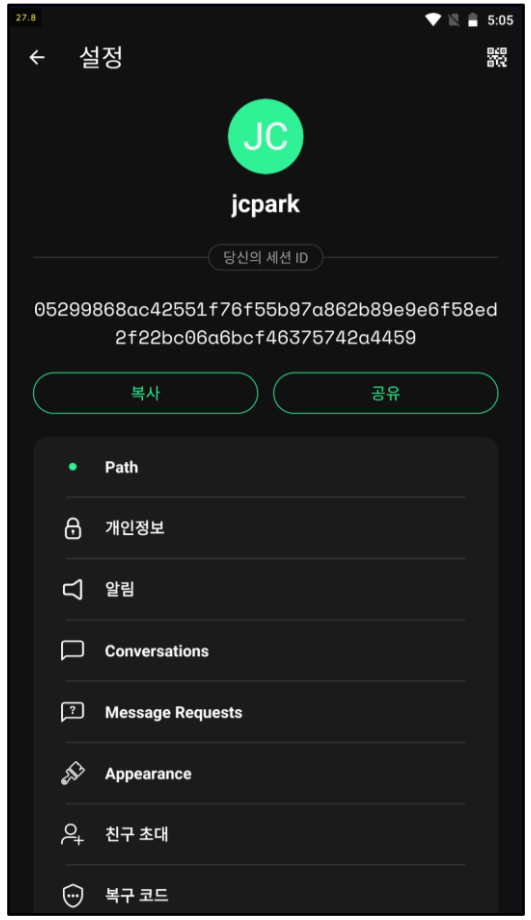


- Oxen Project에서 2020년 개발
- 익명 및 보안을 제공하는 개인 메신저
- 익명 계정생성, 첨부파일 보호 기능

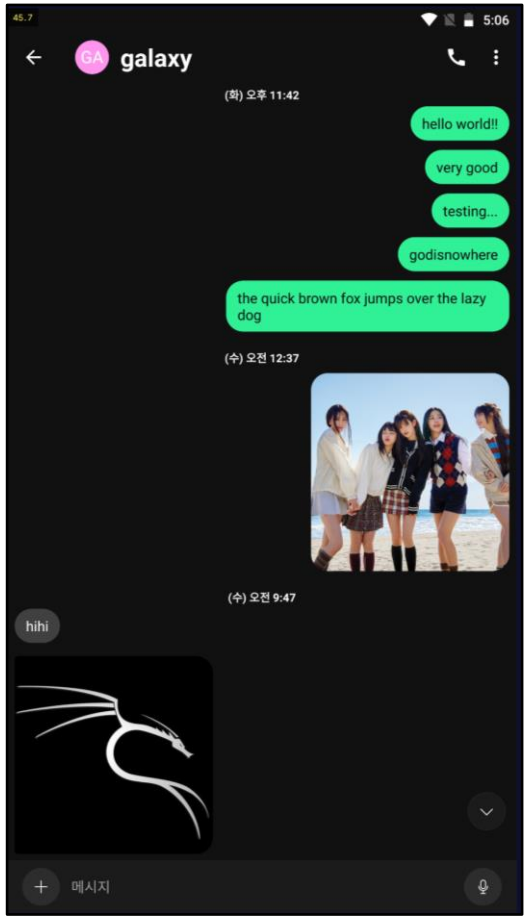
2. 아티팩트 적립과정

2. 아티팩트 적립과정

☞ 녹스에서 진행, 갤럭시 S8+/ Android 7.1.2

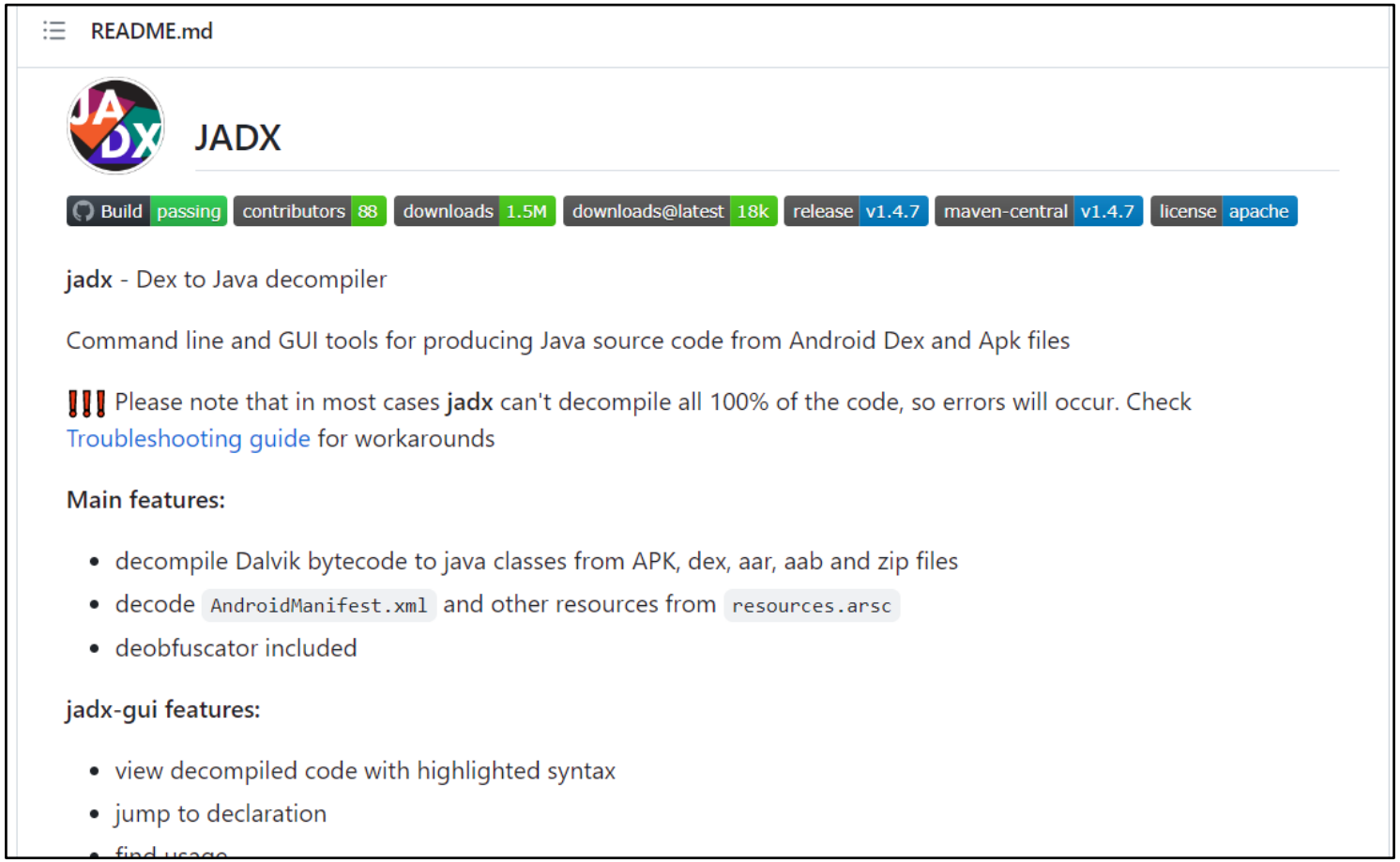


☞개인 계정 생성



☞채팅을 통해 다양한 메시지 전송

2. 아티팩트 적립과정



README.md

JADX

Build passing | contributors 88 | downloads 1.5M | downloads@latest 18k | release v1.4.7 | maven-central v1.4.7 | license apache

jadx - Dex to Java decompiler

Command line and GUI tools for producing Java source code from Android Dex and Apk files

❗❗❗ Please note that in most cases **jadx** can't decompile all 100% of the code, so errors will occur. Check Troubleshooting guide for workarounds

Main features:

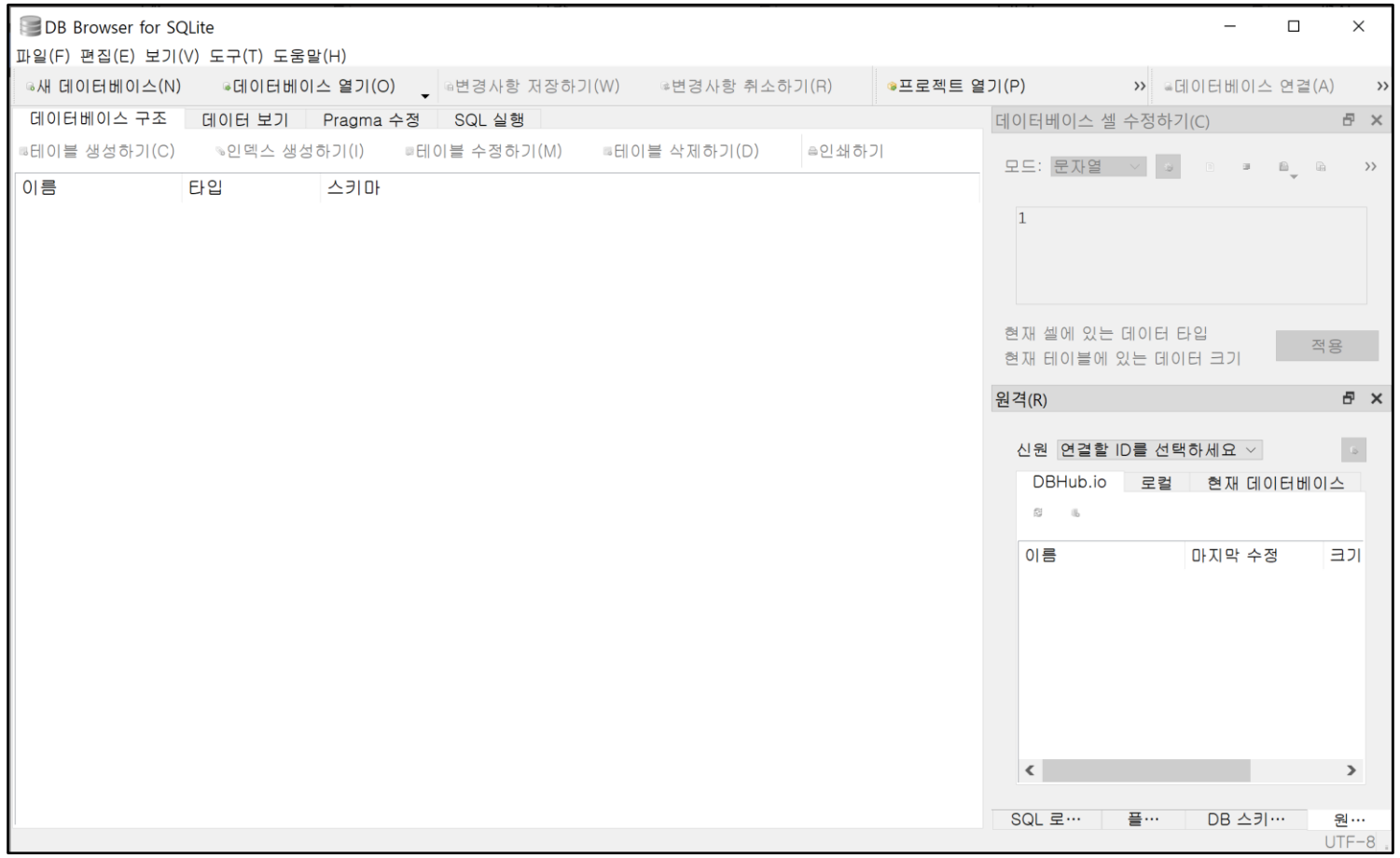
- decompile Dalvik bytecode to java classes from APK, dex, aar, aab and zip files
- decode `AndroidManifest.xml` and other resources from `resources.arsc`
- deobfuscator included

jadx-gui features:

- view decompiled code with highlighted syntax
- jump to declaration
- find usage

👉 jadx를 통해 APK파일 분석

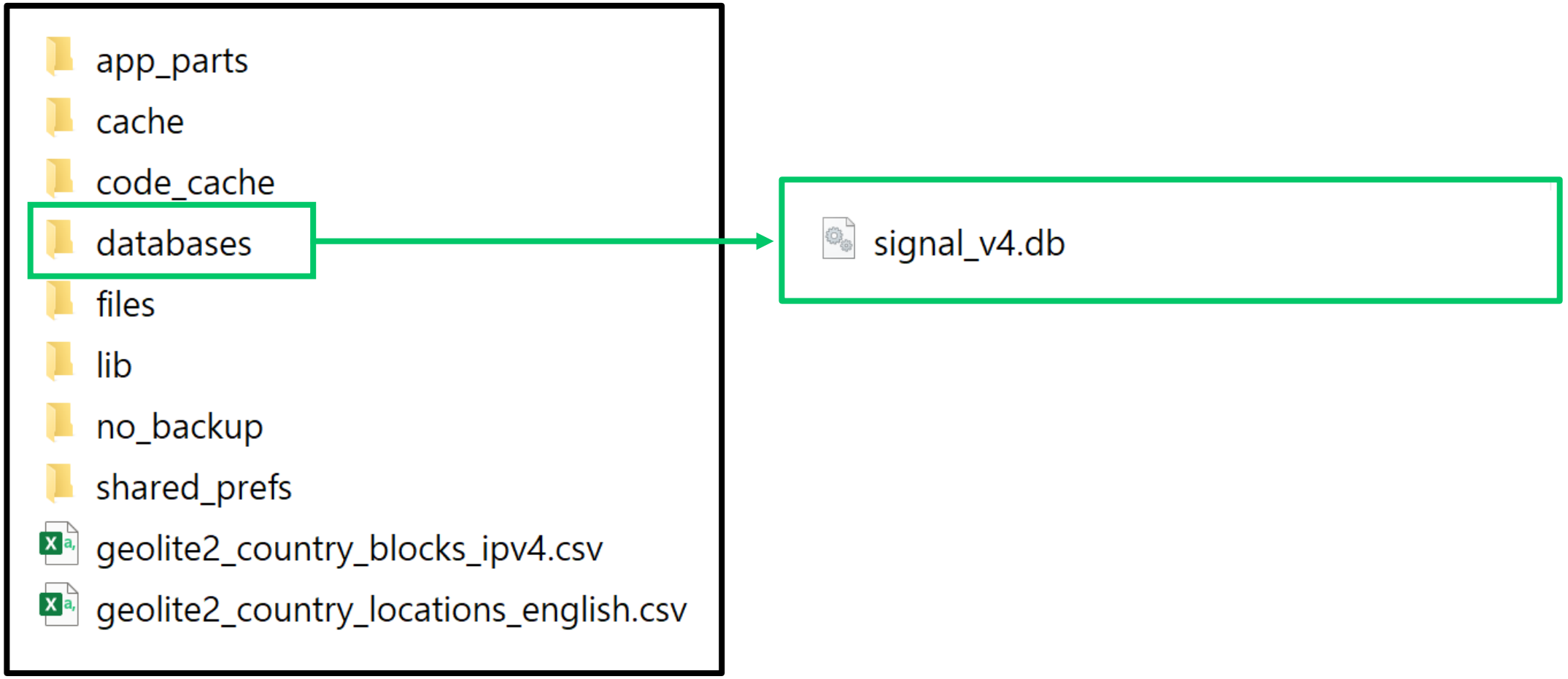
2. 아티팩트 적립과정



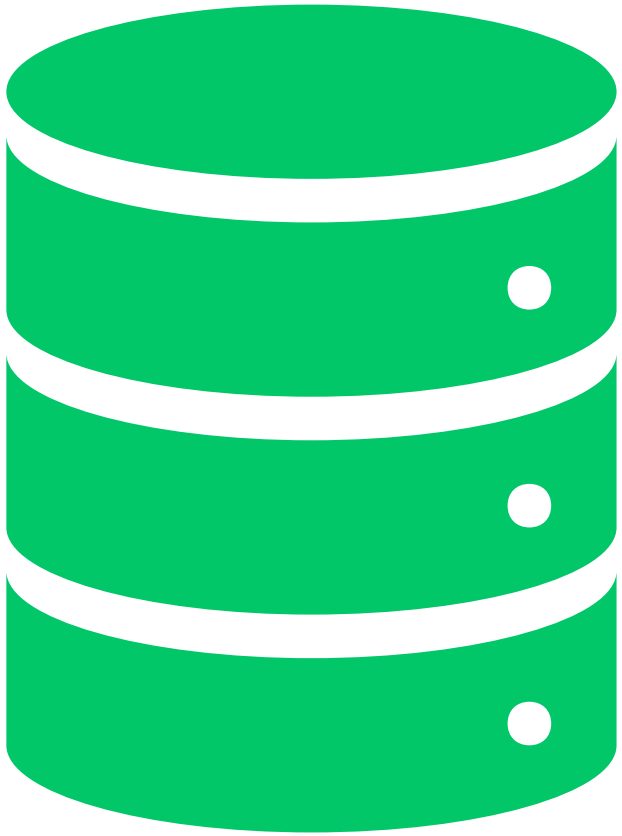
☞ DB Browser를 통해 데이터 베이스 분석

3. 데이터 파일 분석

3. 데이터 파일 분석



👉 databases폴더에 signal_v4.db 데이터베이스 존재



4. 데이터 베이스 복호화

4. 데이터 베이스 복호화

SQLCipherOpenHelper

```
/* loaded from: classes4.dex */
46 public class SQLCipherOpenHelper extends SQLiteOpenHelper {
    private static final String CIPHER3_DATABASE_NAME = "signal.db";
    public static final String DATABASE_NAME = "signal_v4.db";
    private static final int DATABASE_VERSION = 63;
    private static final int MIN_DATABASE_VERSION = 28;
    private static final String TAG = "SQLCipherOpenHelper";
    private static final int lokiV10 = 31;
    private static final int lokiV11 = 32;
    private static final int lokiV12 = 33;
    private static final int lokiV13 = 34;
    private static final int lokiV14_BACKUP_FILES = 35;
    private static final int lokiV15 = 36;
    private static final int lokiV16 = 37;
    private static final int lokiV17 = 38;
    private static final int lokiV18_CLEAR_BG_POLL_JOBS = 39;
    private static final int lokiV19 = 40;
    private static final int lokiV20 = 41;
    private static final int lokiV21 = 42;
    private static final int lokiV22 = 43;
    private static final int lokiV23 = 44;
    private static final int lokiV24 = 45;
    private static final int lokiV25 = 46;
    private static final int lokiV26 = 47;
```

👉 SQLCipherOpenHelper 클래스에서 signal_v4.db 생성

4. 데이터 베이스 복호화

```java
public static void migrateSqlCipher3To4IfNeeded(Context context, DatabaseSecret databaseSecret) throws Exception {
    String path = context.getDatabasePath(CIPHER3_DATABASE_NAME).getPath();
    File file = new File(path);
    if (file.exists()) {
        String path2 = context.getDatabasePath(DATABASE_NAME).getPath();
        File file2 = new File(path2);
        try {
            if (file2.exists()) {
                if (file.lastModified() <= file2.lastModified()) {
                    try {
                        SQLiteDatabase open = open(path2, databaseSecret, true);
                        int version = open.getVersion();
                        open.close();
                        if (version > 0) {
                            return;
                        }
                    } catch (Exception unused) {
                        Log.m207i(TAG, "Failed to retrieve version from new database, assuming invalid and remigrating");
                    }
                }
                if (!file2.delete()) {
                    throw new Exception("Failed to remove invalid new database");
                }
            }
            if (!file2.createNewFile()) {
                throw new Exception("Failed to create new database");
            }
            SQLiteDatabase open2 = open(path, databaseSecret, false);
            int version2 = open2.getVersion();
```

👉 SQLCipherOpenHelper 클래스에서 signal_v4.db 생성

4. 데이터 베이스 복호화

SQLCipherOpenHelper | DatabaseModule | DatabaseSecretProvider | KeyStoreHelper | Texts

```
    @Provides
    @Singleton
33  public final SQLCipherOpenHelper provideOpenHelper(@ApplicationContext Context context) {
        Intrinsics.checkNotNullParameter(context, "context");
34      DatabaseSecret orCreateDatabaseSecret = new DatabaseSecretProvider(context).getOrCreateDatabaseSecret();
35      SQLCipherOpenHelper.migrateSqlCipher3To4IfNeeded(context, orCreateDatabaseSecret);
36      return new SQLCipherOpenHelper(context, orCreateDatabaseSecret);
    }

    @Provides
    @Singleton
40  public final SmsDatabase provideSmsDatabase(@ApplicationContext Context context, SQLCipherOpenHelper openHelper) {
        Intrinsics.checkNotNullParameter(context, "context");
        Intrinsics.checkNotNullParameter(openHelper, "openHelper");
41      return new SmsDatabase(context, openHelper);
    }

    @Provides
    @Singleton
44  public final MmsDatabase provideMmsDatabase(@ApplicationContext Context context, SQLCipherOpenHelper openHelper) {
        Intrinsics.checkNotNullParameter(context, "context");
        Intrinsics.checkNotNullParameter(openHelper, "openHelper");
45      return new MmsDatabase(context, openHelper);
    }
```

👉 migrate…()가 사용된 메소드로 이동

4. 데이터 베이스 복호화

SQLCipherOpenHelper | DatabaseModule | DatabaseSecretProvider | KeyStoreHelper

```java
19      public DatabaseSecretProvider(Context context) {
21          this.context = context.getApplicationContext();
        }

24      public DatabaseSecret getOrCreateDatabaseSecret() {
25          String databaseUnencryptedSecret = TextSecurePreferences.CC.getDatabaseUnencryptedSecret(this.context);
26          String databaseEncryptedSecret = TextSecurePreferences.CC.getDatabaseEncryptedSecret(this.context);
            if (databaseUnencryptedSecret != null) {
28              return getUnencryptedDatabaseSecret(this.context, databaseUnencryptedSecret);
            }
            if (databaseEncryptedSecret != null) {
29              return getEncryptedDatabaseSecret(databaseEncryptedSecret);
            }
30          return createAndStoreDatabaseSecret(this.context);
        }
```

👉 키를 만드는 부분인 getOr…()메소드로 이동

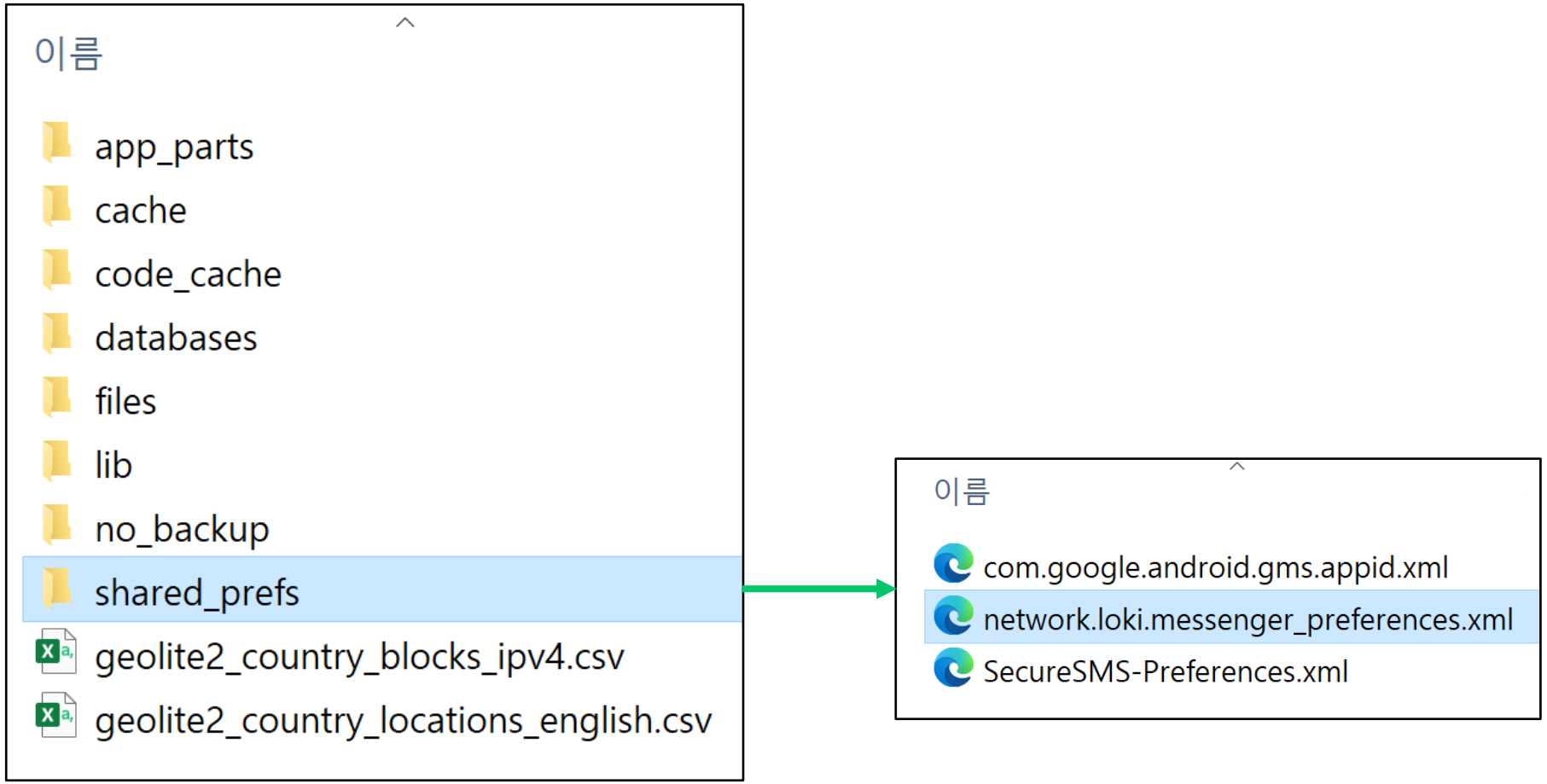
4. 데이터 베이스 복호화

SQLCipherOpenHelper | DatabaseModule | DatabaseSecretProvider | KeyStoreHelper

```
19      public DatabaseSecretProvider(Context context) {
21          this.context = context.getApplicationContext();
        }

24      public DatabaseSecret getOrCreateDatabaseSecret() {
25          String databaseUnencryptedSecret = TextSecurePreferences.CC.getDatabaseUnencryptedSecret(this.context);
26                                                                                                  this.context);
28          @JvmStatic
            public static String getDatabaseUnencryptedSecret(Context context) {
                return TextSecurePreferences.Companion.getDatabaseUnencryptedSecret(context);
            }
29          @JvmStatic
            public final String getDatabaseUnencryptedSecret(Context context) {
30              Intrinsics.checkNotNullParameter(context, "context");
                return getStringPreference(context, "pref_database_unencrypted_secret", null);
            }
        }
```

☞ databaseUnenc…는 pref_database_unenc…의 값을 가짐

4. 데이터 베이스 복호화



👉 pref_database_unenc…는 network.loki….xml에 존재

4. 데이터 베이스 복호화


```
▼<map>
  <string name="pref_fcm_token">fvcMVGoyI-8:APA91bFHCMafWdKo_iSYcDOV1sHcqBPQiCxPoqCJE9mCf3x2JH86m8CL1wxLD96cU11_GZVh1eryCUL5KuOxeuLSyuoYIXqC60haKRJyrQs3wOBK9axlizdHc9e8RHxAitv6WCar-ekX</string>
  <string name="pref_profile_name">jcpark</string>
  <int name="last_version_code" value="354"/>
  <boolean name="pref_configuration_synced" value="true"/>
  <boolean name="pref_is_using_fcm" value="true"/>
  <string name="pref_attachment_encrypted_secret">
  {"data":"Z4mceWOm1AWwe6nWPJ15wRsIPC4PR8ys010gzX1EG79oCCOSVX2TwQsgjjOMtcikhrTEwQcRVihJiDIebhMPUiW5m74RIZ3Cjz32xrhcfcV4rYTec4NMiQAcSFvk/Ov9yGhBFNzGQI7PQqcFAb/53fxETax4vcOg","iv":"FDpOuyvr+KbzML+1"}</string>
  <boolean name="has_viewed_seed" value="true"/>
  <long name="pref_last_profile_update_time" value="0"/>
  <string name="pref_local_number">05299868ac42551f76f55b97a862b89e9e6f58ed2f22bc06a6bcf46375742a4459</string>
  <string name="pref_log_encrypted_secret">{"data":"DNxrjepPu8RAxRmCf5dIsEOuYIyr/ICAeCZUmM4XihdPMMQjnXbvoAJKe9qy8L4O","iv":"9ZAkVHXLd2FRnjM7"}</string>
  <boolean name="pref_disable_passphrase" value="true"/>
  <long name="pref_last_fcm_token_upload_time_2" value="1692714919744"/>
  <long name="restoration_time" value="1691340850368"/>
  <int name="pref_profile_avatar_id" value="0"/>
  <long name="pref_last_vacuum_time" value="1692714914528"/>
  <boolean name="pref_seen_welcome_screen" value="true"/>
  <string name="pref_database_encrypted_secret">{"data":"o+rA3UoMqHHYQGDHT/uL2VthdcI4Hz022NAjbktozfzjhvBE1x5yXvyt5hOB95PU","iv":"HBsSCZPFWQNVcsIx"}</string>
  <int name="pref_local_registration_id" value="11857"/>
  <long name="pref_last_configuration_sync_time" value="1691341541015"/>
</map>
```


👉 pref_database_unenc…는 network.loki….xml에 존재

4. 데이터 베이스 복호화

SQLCipherOpenHelper | DatabaseModule | DatabaseSecretProvider | KeyStoreHelper

```
19      public DatabaseSecretProvider(Context context) {
21          this.context = context.getApplicationContext();
        }

24      private DatabaseSecret getEncryptedDatabaseSecret(String str) {
25          if (Build.VERSION.SDK_INT < 23) {
26              throw new AssertionError("OS downgrade not supported. KeyStore sealed data exists on platform < M!");
            }
28          return new DatabaseSecret(KeyStoreHelper.unseal(KeyStoreHelper.SealedData.fromString(str)));
        }

            if (databaseEncryptedSecret != null) {
29              return getEncryptedDatabaseSecret(databaseEncryptedSecret);
            }
30          return createAndStoreDatabaseSecret(this.context);
        }
```

👉 리턴값이 되는 getEncryptedDatabaseSecret()메소드로 이동

4. 데이터 베이스 복호화

```java
private DatabaseSecret getEncryptedDatabaseSecret(String str) {
    if (Build.VERSION.SDK_INT < 23) {
        throw new AssertionError("OS downgrade not supported. KeyStore sealed data exists on platform < M!");
    }
    return new DatabaseSecret(KeyStoreHelper.unseal(KeyStoreHelper.SealedData.fromString(str)));
}
```

```java
public DatabaseSecret(byte[] bArr) {
    this.key = bArr;
    this.encoded = Hex.toStringCondensed(bArr);
}
```

👉 DatabaseSecret()는 받은 변수를 키로 지정함

4. 데이터 베이스 복호화

```
private DatabaseSecret getEncryptedDatabaseSecret(String str) {
    if (Build.VERSION.SDK_INT < 23) {
        throw new AssertionError("OS downgrade not supported. KeyStore sealed data exists on platform < M!");
    }
    return new DatabaseSecret(KeyStoreHelper.unseal(KeyStoreHelper.SealedData.fromString(str)));
}
```

👉 키로 지정되는 unseal()메소드로 이동

4. 데이터 베이스 복호화

```java
public static byte[] unseal(SealedData sealedData) {
    byte[] doFinal;
    SecretKey keyStoreEntry = getKeyStoreEntry();
    try {
        synchronized (CipherUtil.CIPHER_LOCK) {
            Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding");
            cipher.init(2, keyStoreEntry, new GCMParameterSpec(128, sealedData.f990iv));
            doFinal = cipher.doFinal(sealedData.data);
        }
        return doFinal;
    } catch (InvalidAlgorithmParameterException | InvalidKeyException | NoSuchAlgorithmException | BadPaddingException | IllegalBlockSizeException | NoSuchPaddingException e) {
        throw new AssertionError(e);
    }
}
```

👉 키로 지정되는 unseal()메소드로 이동

4. 데이터 베이스 복호화

```java
public static byte[] unseal(SealedData sealedData) {
    byte[] doFinal;
    SecretKey keyStoreEntry = getKeyStoreEntry();
    try {
        synchronized (CipherUtil.CIPHER_LOCK) {
            Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding");
            cipher.init(2, keyStoreEntry, new GCMParameterSpec(128, sealedData.f990iv));
            doFinal = cipher.doFinal(sealedData.data);
        }
        return doFinal;
    } catch (InvalidAlgorithmParameterException | InvalidKeyException | NoSuchAlgorithmException | BadPaddingException | IllegalBlockSizeException | NoSuchPaddingException e) {
        throw new AssertionError(e);
    }
}
```

👉 keyStoreEntry를 키로 하여 AES-GCM으로 암호화한 값을 출력

4. 데이터 베이스 복호화

```java
public static byte[] unseal(SealedData sealedData) {
    byte[] doFinal;
    SecretKey keyStoreEntry = getKeyStoreEntry();
    try {
        synchronized (CipherUtil.CIPHER_LOCK) {
            Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding");
            cipher.init(2, keyStoreEntry, new GCMParameterSpec(128, sealedData.f990iv));
            doFinal = cipher.doFinal(sealedData.data);
        }
        return doFinal;
    } catch (InvalidAlgorithmParameterException | InvalidKeyException | NoSuchAlgorithmException | BadPaddingException | IllegalBlockSizeException | NoSuchPaddingException e) {
        throw new AssertionError(e);
    }
}
```

👉 keyStoreEntry를 만드는 getKeyStoreEntry()메소드로 이동

4. 데이터 베이스 복호화

```java
private static SecretKey getKeyStoreEntry() {
    KeyStore keyStore = getKeyStore();
    try {
        try {
            return getSecretKey(keyStore);
        } catch (UnrecoverableKeyException e) {
            throw new AssertionError(e);
        }
    } catch (UnrecoverableKeyException unused) {
        return getSecretKey(keyStore);
    }
}
```

👉 keyStoreEntry를 만드는 getKeyStoreEntry()메소드로 이동

4. 데이터 베이스 복호화

```java
public final class KeyStoreHelper {
    private static final String ANDROID_KEY_STORE = "AndroidKeyStore";
    private static final String KEY_ALIAS = "SignalSecret";
```

```java
private static SecretKey getKeyStoreEntry() {
    KeyStore keyStore = getKeyStore();
    try {
```

```java
private static KeyStore getKeyStore() {
    try {
        KeyStore keyStore = KeyStore.getInstance(ANDROID_KEY_STORE);
        keyStore.load(null);
        return keyStore;
    } catch (IOException | KeyStoreException | NoSuchAlgorithmException | CertificateException e) {
        throw new AssertionError(e);
    }
}
```

👉 keyStore에는 AndroidKeyStore의 값이 들어감

4. 데이터 베이스 복호화

```java
public final class KeyStoreHelper {
    private static final String ANDROID_KEY_STORE = "AndroidKeyStore";
    private static final String KEY_ALIAS = "SignalSecret";
```

```java
private static SecretKey getKeyStoreEntry() {
    KeyStore keyStore = getKeyStore();
    try {
        try {
            return getSecretKey(keyStore);
        } catch (UnrecoverableKeyException e) {
            throw new AssertionError(e);
        }
    } catch (UnrecoverableKeyException unused) {
        return getSecretKey(keyStore);
    }
}
```

```java
private static SecretKey getSecretKey(KeyStore keyStore) throws UnrecoverableKeyException {
    try {
        return ((KeyStore.SecretKeyEntry) keyStore.getEntry(KEY_ALIAS, null)).getSecretKey();
    } catch (KeyStoreException e) {
        e = e;
        throw new AssertionError(e);
    } catch (NoSuchAlgorithmException e2) {
        e = e2;
        throw new AssertionError(e);
    } catch (UnrecoverableKeyException e3) {
        throw e3;
    } catch (UnrecoverableEntryException e4) {
        e = e4;
        throw new AssertionError(e);
    }
}
```

👉 리턴값으로 signalSecret라는 이름의 안드로이드키를 사용

4. 데이터 베이스 복호화

SQLCipherOpenHelper | DatabaseModule | DatabaseSecretProvider | KeyStoreHelper

```java
19      public DatabaseSecretProvider(Context context) {
21          this.context = context.getApplicationContext();
        }

24      public DatabaseSecret getOrCreateDatabaseSecret() {
25          String databaseUnencryptedSecret = TextSecurePreferences.CC.getDatabaseUnencryptedSecret(this.context);
26          String databaseEncryptedSecret = TextSecurePreferences.CC.getDatabaseEncryptedSecret(this.context);
            if (databaseUnencryptedSecret != null) {
28              return getUnencryptedDatabaseSecret(this.context, databaseUnencryptedSecret);
            }
            if (databaseEncryptedSecret != null) {
29              return getEncryptedDatabaseSecret(databaseEncryptedSecret);
            }
30          return createAndStoreDatabaseSecret(this.context);
        }
```

```java
private DatabaseSecret createAndStoreDatabaseSecret(Context context) {
    byte[] bArr = new byte[32];
    new SecureRandom().nextBytes(bArr);
    DatabaseSecret databaseSecret = new DatabaseSecret(bArr);
    if (Build.VERSION.SDK_INT >= 23) {
        TextSecurePreferences.CC.setDatabaseEncryptedSecret(context, KeyStoreHelper.seal(databaseSecret.asBytes()).serialize());
    } else {
        TextSecurePreferences.CC.setDatabaseUnencryptedSecret(context, databaseSecret.asString());
    }
    return databaseSecret;
}
```

☞ 데이터에서 32바이트만 쓰임

4. 데이터 베이스 복호화

👉 데이터베이스 키 생성

- 데이터: pref_database_encrypted_secret의 data의 32바이트 값
- 키: SignalSecret이라는 안드로이드 키
- iv: pref_database_encrypted_secret의 iv 값

4. 데이터 베이스 복호화

```
from Crypto.Cipher import AES
import base64

encrypted_secret_data=base64.b64decode("o+rA3UoMqHHYQGDHT/uL2VthdcI4HzO22NAjbktozfzjhvBE1x5yXvyt5hOB95PU".encode())
encrypted_secret_iv=base64.b64decode("HBsSCZPFWQNVcsIx".encode())
Key=b'\x3F\x9F\x47\xA9\xDB\x8F\x51\x23\xB1\xD1\x85\xA7\x5B\x01\x93\xE1'

data=encrypted_secret_data[:32]
iv=encrypted_secret_iv

cipher=AES.new(Key, AES.MODE_GCM, iv)
dec_key=cipher.decrypt(data)

sql_key=dec_key.hex()
print(sql_key)
```

👉 키 생성 파이썬 코드 작성

4. 데이터 베이스 복호화

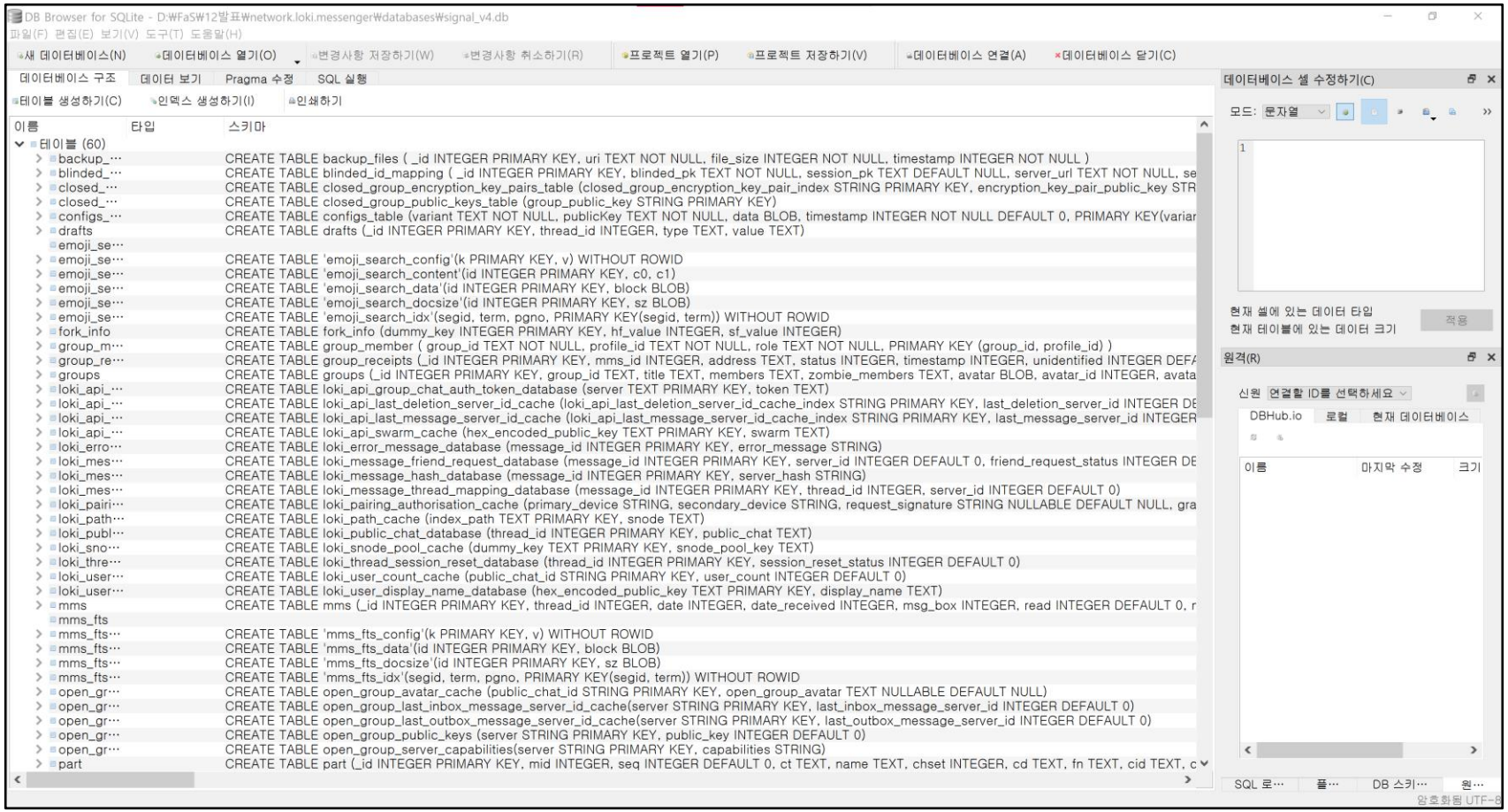
```
from Crypto.Cipher import AES
import base64

encrypted_secret_data=base64.b64decode("o+rA3UoMqHHYQGDHT/uL2VthdcI4HzO22NAjbktozfzjhvBE1x5yXvyt5hOB95PU".encode())
encrypted_secret_iv=base64.b64decode("HBsSCZPFWQNVcsIx".encode())
Key=b'\x3F\x9F\x47\xA9\xDB\x8F\x51\x23\xB1\xD1\x85\xA7\x5B\x01\x93\xE1'

data=encrypted_secret_data[:32]
iv=encrypted_secret_iv

cipher=AES.new(Key, AES.MODE_GCM, iv)
dec_key=cipher.decrypt(data)

sql_key=dec_key.hex()
print(sql_key)
```

↓

```
91253ed215b5651cd837aaeccd6b3a9581e8d9267ae8f8f1f0497efa7e7916b6
```

👉 키 생성 파이썬 코드 작성

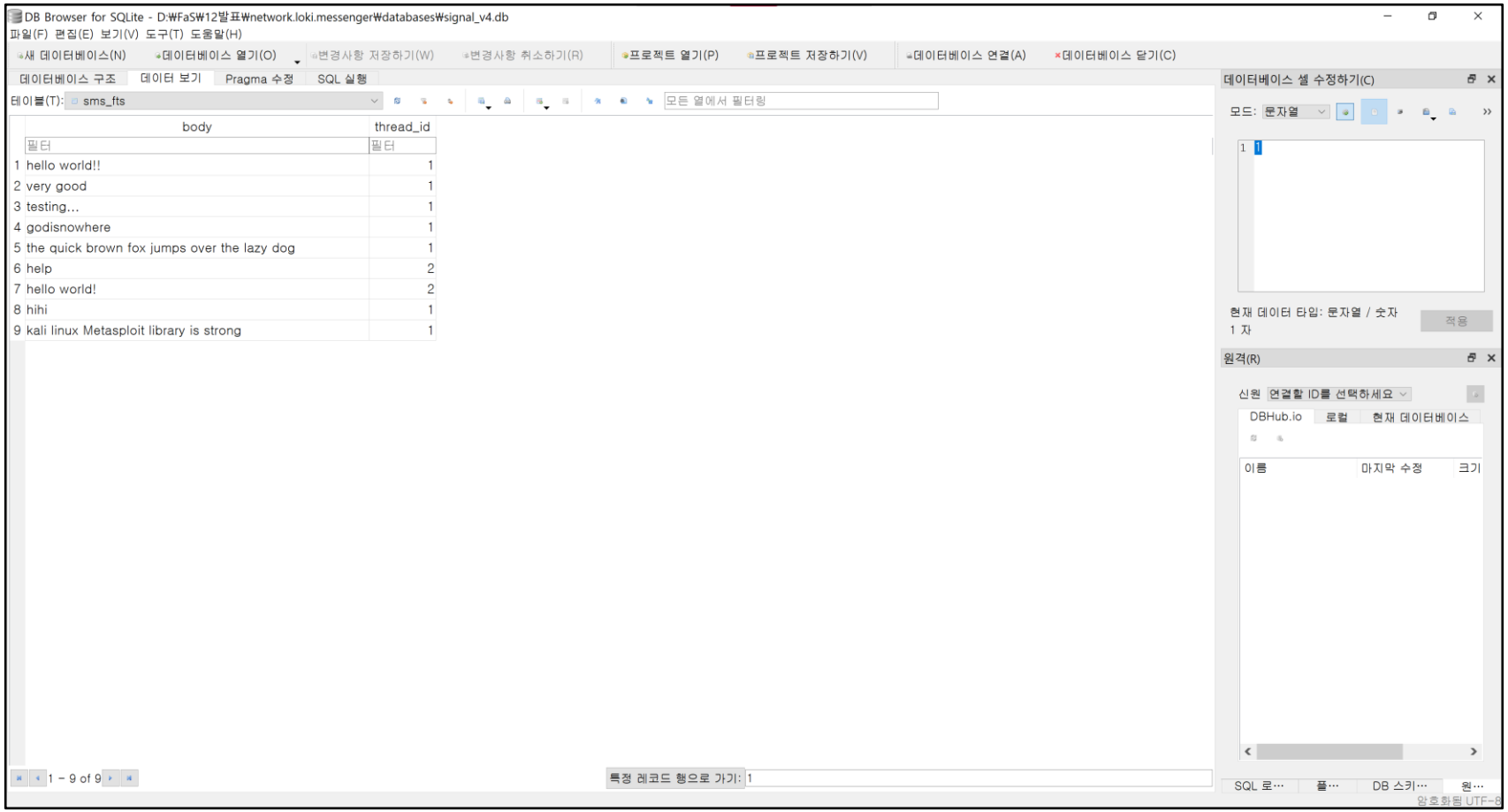
4. 데이터 베이스 복호화



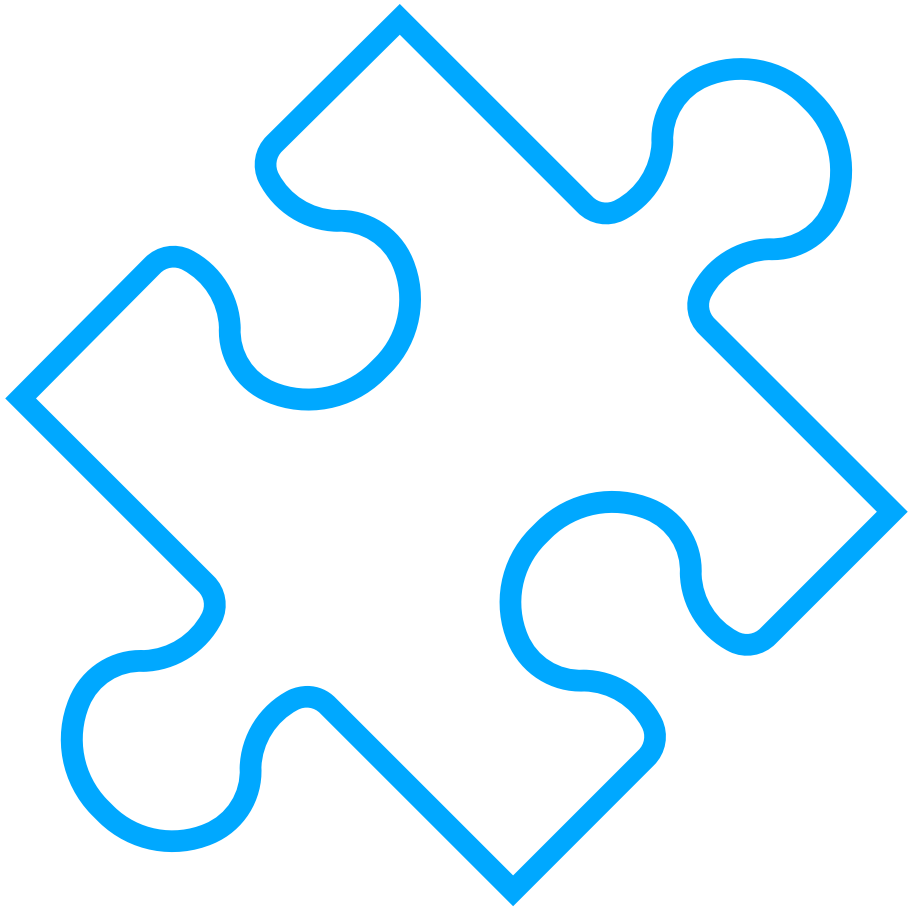
☞ 데이터 베이스 복호화 성공

5. 복호화된 메시지 확인

5. 복호화된 메시지 확인



☞ sms_fts 테이블에서 메시지 확인 가능



6. 해결하지 못한 부분

6. 해결하지 못한 부분

👉 보내고 받은 파일의 복호화는 아직 해결하지 못함

데이터베이스 구조 | 데이터 보기 | Pragma 수정 | SQL 실행

테이블(T): part

	_id	mid	seq	ct	name	chset	cd	fn	cid	cl	ctt_s	ctt_t	encrypted	pending_push	_data	data_size	file_name	thumbnail
1	1	1	0	image/jpeg	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0	/data/user/0/···	49687		NULL
2	2	2	0	image/jpeg	NULL	NULL	w9vwNGLZuNKJ···	NULL	NULL	4456703834929···	NULL	NULL	NULL	0	/data/user/0/···	2678899	NewJeans_zero ···	/data/user/0/···

👉 part 테이블에서 파일의 정보는 확인 가능

6. 해결하지 못한 부분

👉 보내고 받은 파일의 복호화는 아직 해결하지 못함

테이블(T): part

	cl	ctt_s	ctt_t	encrypted	pending_push	_data	data_size	file_name	thumbnail
		필터	필터	필터	필터	필터	필터	필터	필터
1		NULL	NULL	NULL	0	/data/user/0/network.loki.messenger/app_parts/part1299718442.mms	49687		NULL
2	703834929···	NULL	NULL	NULL	0	/data/user/0/network.loki.messenger/app_parts/part596267436.mms	2678899	NewJeans_zero ···	/data/user/0/···

👉 _data에서 데이터가 있을 것으로 추정되는 mms 파일 확인

6. 해결하지 못한 부분

- 👉 session은 signal 메신저를 기반으로 만들어진 애플리케이션
  - 👉 현재 signal의 첨부파일 복호화 방법을 찾는 중…
  - 👉 jadx를 통해 mms 파일이 만들어지는 과정이 보이는지 찾는 중…

THANKS TO WATCHING

20192233 박진철